s1061443_AIdea

Rank

Public Leaderboard | Private Leaderboard

排名	隊伍名稱	成績	上傳時間	次數
1	cychung	5.5180281	2021/05/06 20:19:57	46
2	peter0512lee	5.8606596	2021/05/12 23:28:09	9
3	chiangEric	5.8975308	2021/05/13 22:53:09	2
4	love1245672	6.0305054	2021/05/13 15:20:00	120
5	PoHuaChen	6.1131382	2021/05/18 14:01:29	6
6	yang1114	6.3840382	2021/05/21 08:14:42	14
7	s1061221	6.8390596	2021/05/22 11:07:38	9
8	mitch_hsu	6.9688942	2021/05/07 13:57:25	7
9	phil_wang	7.0115217	2021/05/20 11:10:22	8
10	yzu1061504	7.0991694	2021/05/15 12:08:25	5
11	qpchung	7.1193372	2021/05/22 08:34:39	48
12	howardq12q	7.2661792	2021/05/22 01:33:11	1
13	miona	7.4798979	2021/05/14 15:19:13	1
14	hongweijay	7.4801871	2021/05/24 16:02:01	17
15	naiqi97	7.5681048	2021/05/19 17:05:27	2
16	liuuuu	7.5847911	2021/05/28 23:33:58	1

Code Description

Data preprocessing

把train中乳量是空的刪掉

```
train = train.dropna(subset=['11'])
train.reset_index(drop=True, inplace=True)
```

將乳牛的空值填入平均體重

```
avg_weight = birth['6'].mean()
birth['6'] = birth['6'].fillna(birth['6'].mean())
```

train 合併 spec, 當年當月有病1, 沒病0, 新增health欄位

```
from datetime import datetime
train['health'] = 0
test['health'] = 0
for i in range(len(spec)):
    ym = datetime.strptime(spec['4'][i], "%Y/%m/%d %H:%M")
    if len(train.index[train["5"] == spec["1"][i]]) > 0:
        for j in train.index[train["5"] == spec["1"][i]]:
            if train['2'][j] == ym.year and train['3'][j] == ym.month and train['4']
[j]==spec['7'][i]:
                train['health'][j] = 1
    if len(test.index[test["5"] == spec["1"][i]]) > 0:
        for j in test.index[test["5"] == spec["1"][i]]:
            if test['2'][j] == ym.year and test['3'][j] == ym.month and test['4']
[j]==spec['7'][i]:
                test['health'][j] = 1
```

新增weight欄位

```
train['weight'] = np.nan
test['weight'] = np.nan
for i in range(len(birth)):
    if len(train.index[train['5'] == birth['1'][i]])>0:
        for j in train.index[train['5'] == birth['1'][i]]:
                train['weight'][j] = birth['6'][i]
    if len(test.index[test['5'] == birth['1'][i]])>0:
        for j in test.index[test['5'] == birth['1'][i]]:
                test['weight'][j] = birth['6'][i]
train['weight'] = train['weight'].fillna(avg_weight)
test['weight'] = test['weight'].fillna(avg_weight)
```

新增season欄位

```
train['season'] = ""
for index, row in train.iterrows():
    if int(train['3'][index]) >= 3 and int(train['3'][index]) <= 5:
        train['season'][index] = 'Spring'
    elif int(train['3'][index]) >= 6 and int(train['3'][index]) <= 8:
```

```
        train['season'][index] = 'Summer'
    elif int(train['3'][index]) >= 9 and int(train['3'][index]) <= 11:
        train['season'][index] = 'Autumn'
    else:
        train['season'][index] = 'Winter'

test['season'] = ""
for index, row in test.iterrows():
    if int(test['3'][index]) >= 3 and int(test['3'][index]) <= 5:
        test['season'][index] = 'Spring'
    elif int(test['3'][index]) >= 6 and int(test['3'][index]) <= 8:
        test['season'][index] = 'Summer'
    elif int(test['3'][index]) >= 9 and int(test['3'][index]) <= 11:
        test['season'][index] = 'Autumn'
    else:
        test['season'][index] = 'Winter'
```

DNN Model

把要 one hot 的類別轉換成數字

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
all_data=pd.concat([new_train,new_test])
all_data['4'] = labelencoder.fit_transform(all_data['4'])
all_data['5'] = labelencoder.fit_transform(all_data['5'])
all_data['season'] = labelencoder.fit_transform(all_data['season'])
all_data['health'] = labelencoder.fit_transform(all_data['health'])
new_train = all_data[0:len(new_train)]
new_test = all_data[len(new_train)::]
all_data=pd.concat([new_train,new_test])
```

把要的類別轉換成 one hot

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
enc.fit(all_data)
X = enc.transform(new_train).toarray()
X_test = enc.transform(new_test).toarray()
print(X.shape, X_test.shape)
```

train, test 切開

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

Define RMSE Loss

```
from keras import backend as K
def rmse(y_pred, y_true):
    return K.sqrt(K.mean(K.square(y_pred - y_true)))
```

Model design

```
from keras.models import Sequential
from keras.layers import Dense,Dropout,BatchNormalization
from keras.optimizers import Adam

model=Sequential()
model.add(Dense(256, input_dim=3098, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1))
model.compile(loss=rmse, optimizer="adam", metrics=[rmse])
```

Model summary

```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_14 (Dense)             (None, 256)               793344
_________________________________________________________________
dense_15 (Dense)             (None, 256)               65792
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_16 (Dense)             (None, 1)                 257
=================================================================
Total params: 859,393
Trainable params: 859,393
Non-trainable params: 0
_________________________________________________________________
```

Authors

- @peter0512lee